Une nouvelle approche de la carte à puce réseau

Pascal URIEN, P.Urien@frlv.bull.fr
Bull CP8 - R&D 68, Route de Versailles 78431 Louveciennes Cedex

Hayder SALEH, H.Saleh@frlv.bull.fr
Doctorant laboratoire Prisme – Versailles



Nous avons défini et réalisé un concept de carte à puce internet. Une carte à puce internet est un dispositif capable de se comporter comme un nœud du réseau internet et qui implémente des applications serveurs ou clientes (spécifiées par des standards internet, tels que RFC 2068 http 1.1,…). Une carte à puce est une fine tranche de silicium qui comporte un CPU et de la mémoire, et dont le seul lien avec le monde extérieur est une liaison série. Une nouvelle architecture de communication a été étudiée pour l'ensemble terminal carte, cette pile protocolaire permet à la carte de partager les ressources réseaux disponibles sur le terminal. Nous avons démontré la faisabilité de ce concept sur une carte Java associée à un terminal win32. Notre première carte internet réalise un serveur web et un proxy qui renforcent la sécurité des accès web.

Introduction

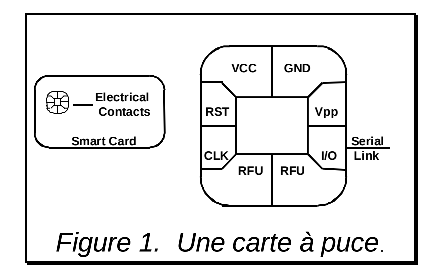


Figure 1. Une carte à puce.

Depuis plusieurs années Bull CP8 mène des recherches sur une carte à puce spécifiquement dédiées au réseau internet. L'idée de base est de considérer la carte comme un ***pocket network computer*** capable d'utiliser les ressources du terminal auquel elle est reliée (clavier écran souris navigateur accès internet). La carte se comporte comme un nœud véritable du réseau internet, elle réalise des applications conformes aux standards internet (RFC - Request For Comment, http, courrier électronique...). Par exemple elle comporte un serveur web accessible depuis un navigateur logé sur le terminal, elle est en mesure de gérer plusieurs sessions TCP simultanées (client ou serveur), ce qui lui permet de réaliser un proxy qui agit comme une passerelle de sécurité entre le navigateur et un serveur web distant. Nous rentrons dans l'ère de ***l'ubiquitous computing*** ([18] omniprésence des ordinateurs). Ce qui signifie que de plus en plus d'objets courants intègrent un microprocesseur, et ont la capacité de se connecter à internet (selon *Frost & Sullivan* 40 % des dispositifs connectés à internet en 2001 ne seront pas des ordinateurs personnels). Les terminaux internet (multimédia par nature, puisque incorporant un navigateur) tentent de se banaliser. Un utilisateur (parfois mobile) utilisera plusieurs types de terminaux reliés à internet dont il ne sera pas toujours le propriétaire, par exemple :

- Le téléphone mobile, le réseau GPRS permettra l'accès à internet depuis un mobile en France vers le début 2000, mais des solutions propriétaires existent déjà (Nokia...).
- Les consoles de jeux (Dreamcast de Sega...), ou la télévision munie d'un set top box (Netgem…).
- Les organisateurs (Psion, Palm, Pilot VII...), ou les mini portables.

Dans ce contexte la carte authentifie un utilisateur mobile qui accède à une pluralité de terminaux, qui devront si nécessaire être ***configurés*** de manière dynamique pour autoriser l'accès et l'usage d'un service particulier.

Un aperçu de la carte à puce

La carte à puce à microprocesseur (*SPOM* - **S**elf **P**rogrammable **O**ne-chip **M**icrocomputer) a été inventée à la fin des années 70 par Michel Ugon [10],[11][12]. Le groupement des cartes bancaires *CB* crée en 1985 [16] a permis la diffusion de quelques 24 millions de cartes de paiement bancaires (cash card). L'utilisation de ces objets c'est depuis diversifiée, porte-monnaie électronique (pme, electronic purse) transport ([2] cartes sans contacts, *contactless card*), GSM ([3],[4] cartes *SIM*), Santé (carte *sésame vitale*), réseaux informatiques (cartes *RSA*). On estime un marché de un milliard de cartes à puce pour l'an 2000. Aujourd'hui la technologie des cartes repose sur des processeurs 8 bits associés à des capacités mémoires de l'ordre de quelques dizaine de Ko. La taille des puces est limitée (#25 mm2) par la flexibilité du support plastique (PVC) [16]. Une nouvelle génération

de processeur RISC 32 bits est en cours de définition et verra le jour au début du prochain millénaire [23]. Outre la sécurité physique qui s'appuie sur le fait qu'il est impossible de lire le code ou les données stockées dans la puce, la carte fournit de manière traditionnelle deux types de services :

– des algorithmes de chiffrement et le stockage des clés associées, ces algorithmes permettent de réaliser des opérations d'authentification, de signature, de chiffrement...
– des données enregistrées sous forme de fichiers protégés en lecture écriture par différentes méthodes d'authentifications.

■ L'approche classique carte à puce réseaux

Cette vision traditionnelle de la carte se retrouve dans les applications réseaux, peut être parce que les cartes ne sont pas conçues à la base pour de tel environnement. La carte est un élément de sécurité rapporté, non spécifique aux réseaux (internet en particulier). Les fonctions classiques de sécurité dans un contexte réseau sont l'identification (login), l'authentification (mot de passe, secret partagé...), la confidentialité (chiffrement), l'intégrité des données (empreinte de données *digest*) et la non répudiation (clés publiques). Par exemple la carte est insérée dans une architecture sécurisée par Secure Socket Layer (SSL) grâce à des modifications logicielles (DLLs...) du système hôte qui comporte un navigateur, et également du serveur ([21],[22], ISI **I**BM **S**martCard **I**dentification Protocol). Elle permet d'authentifier (signature...) un utilisateur nomade (étudiant...) et de lui affecter des droits en conséquence. De manière plus générale les applications réseaux (authentification d'un utilisateur mobile, commerce électronique) qui mettent en œuvre une carte répondent à un modèle du type ***fat client*** [17], c'est-à-dire que chaque application requiert un logiciel spécifique dédié. En quelques sorte elle est associée à un ***mode d'emploi*** particulier (nommé ***Application Protocol*** dans la terminologie Open Card Framework [6]), qui connaît ses règles d'utilisation [24][25]. Ce modèle engendre des difficultés de gestion lorsque une application peut être mise en œuvre depuis un parc de terminaux hétérogènes (PC, machine unix, portable GSM, borne internet...).

■ Une nouvelle approche

On le constate l'approche du ***fat client*** est un frein pour le développement d'applications réseaux ou un utilisateur nomade utilisent plusieurs terminaux (PC portable, GSM mobile, station de travail fixe...) capables d'accéder à internet. Par exemple si une application est un abonnement électronique à un journal, il semble légitime que le bénéficiaire puisse lire ce journal depuis son bureau, son domicile, dans un hôtel ou à partir de son téléphone mobile. Notre nouvelle approche de la carte réseau consiste à l'adapter à chaque terminal au moyen d'un protocole ***unique*** que nous souhaitons standardiser. Ce protocole est plus particulièrement adapté au support des protocoles de type TCP/IP. Ces nouveaux concepts s'appuient largement sur la technologie Java Card, introduite en 1997 [19], qui permet de définir des architectures sécurisées et ouvertes (la carte n'est plus dédiée à une application spécifique). Notre technologie constitue une révolution de l'ergonomie de la carte et de son usage à travers le réseau internet :

- Le terminal internet est banalisé par un protocole d'adaptation unique, l'utilisateur nomade peut mettre en œuvre les droits associés à sa carte sur un terminal quelconque.
- La carte réalise les protocoles internet, ce n'est plus un élément étranger au réseau mais un nouvel objet de la communauté internet.

La convergence du monde de la téléphonie traditionnelle et de celui des réseaux [18], la disponibilité de terminaux bon marché (type internet multimédia) sont autant de facteurs qui encouragent l'apparition d'***objets virtuels*** que l'on pourra posséder à travers le réseau en l'absence de support physiques (logiciels, multimédia...). Notre approche révolutionnaire de la carte à puce réseau, pièce essentielle de la sécurité requise par la notion de propriété est un premier pas vers ce type de modèle. La carte à puce est une tranche de silicium qui embarque une unité centrale [16], et de la mémoire (ROM, RAM, E2PROM); elle est capable de stocker une quantité modeste de données et de code (de l'ordre de 64 ko). Le seul lien avec le monde extérieur est une liaison série dont la vitesse est comprise entre 9600 et 105900 bauds. La puce ne comporte qu'un seule patte d'entrée sortie (Fig. 1), ce qui implique que seuls des protocoles d'échange de données half-duplex (alternés) peuvent être mis en œuvre. La différence essentielle entre un système informatique ordinaire et une carte est justement que cette dernière ne possède aucun périphérique classique d'entrée sortie [26] comme par exemple un clavier, une souris, un écran ou une carte de communication ethernet). Un *SPOM* a donc besoin de ressources externes pour fonctionner, certaines sont assurées par un lecteur de carte (alimentation en énergie et en horloge) et d'autres fonctions d'entrée/sortie telles que clavier ou écran sont localisées sur le terminal. Aujourd'hui l'interface utilisateur la plus largement répandue est un navigateur web, la disponibilité d'un tel logiciel suppose que le terminal possède une pile protocolaire TCP/IP et soit configuré pour accéder à internet. Notre approche permet à une carte à puce

de partager les ressources réseaux disponibles sur le terminal (pile TCP/IP ou carte réseau), qui par ailleurs ne rajoutent aucune fonction de sécurité. Une carte à puce internet peut donc accéder de manière sécurisée à des quantités de données quasi illimitées, nous pensons qu'elle sera le moteur de nouveaux services tout en s'adaptant à ceux qui existent déjà, citons par exemple :

- ***Sécurité réseau et commerce électronique,*** mise en œuvre de fonctions de sécurité telles que authentification, signature, confidentialité etc…, à travers une connexion entre une carte et un serveur.
- ***Virtual Home***, l'utilisateur nomade reconstitue (télécharge) tout ou partie de son environnement dans un terminal banalisé (non sûr), à l'aide des droits détenus par la carte.
- ***Environnement Virtuel Distribué***, si les performances du réseau le permettent il n'est pas nécessaire de télécharger la totalité d'une application, une utilisation à distance est possible (déport d'écran...), au moyen d'une licence localisée dans la carte (*logiciels virtuels*).
- ***Objets Virtuels***, les objets virtuels sont des objets digitaux que l'on peut posséder sans support physique (objets multimédia tels que CD, films, journaux...). Un droit contenu dans la carte permet la jouissance de l'objet sur tout terminal adapté.
- ***Communauté Virtuelle***, un ensemble de services liés à la connexion simultanée et authentifiée d'un ensemble de cartes capables de communiquer entre elles (messagerie, forum, administration de réseau...).



■ Réseaux et carte proactives

Le standard GSM 11.14 a introduit la notion de commandes proactives. La carte supervise l'exécution de procédures sur le terminal. Par exemple une carte SIM toolkit exécute un jeu et utilise des commandes proactives pour afficher des caractères ou lire des données depuis le clavier du téléphone mobile. Les cartes actuelles sont mises en œuvre d'une manière passive, le terminal contrôle toutes les actions menées par la carte. La proactivité pour une carte internet, signifie qu'un tel dispositif est capable d'accéder au web de sa propre initiative, il réalise des opérations qui ne sont pas prédéfinies par le terminal, en d'autres termes la carte se comporte comme un client TCP.

■ L'architecture TCP/IP

La carte ne possède pas de ressources physiques permettant d'accéder au réseau (carte ethernet, modem...), l'idée de base est l'utilisation depuis la carte des interfaces (logicielles) de communication disponibles sur le système hôte. De fait la même démarche est mise en œuvre lorsqu'un navigateur surfe sur le web, l'application (le navigateur) est capable d'utiliser les ressources réseaux du système hôte sur lequel elle s'exécute. L'utilisation du réseau par une application réseau se résume à la mise en œuvre de couches logicielles fournies par le système hôte, dont l'application connaît un point (ou une méthode) d'accès. Dans l'architecture TCP/IP une application réseau (telle que un navigateur web par exemple) s'appuie sur un modèle qui comporte les 4 premières couches du modèle OSI [8]. Les couches 1 et 2 représentent par exemple la carte (physique) ethernet ou un modem. La machine voit (utilise) une ressource réseau (carte...) à travers une couche d'interface logicielle particulière encore nommée ***driver couches basses*** (ainsi NDIS est une interface couche basse développée par Microsoft). De fait, utiliser une carte réseau dans un système c'est mettre en œuvre un driver couches basses, en terminologie ISO cette interface peut être vue comme un SAP (Service Access Point) de niveau 2 (***SAP2***). De façon analogue une application utilise les couches TCP/IP au moyen d'une ***bibliothèque réseau*** fournie par le système hôte, une application réseau sait mettre en œuvre une telle bibliothèque à travers des SAP de niveaux 3 (SAP3) ou 4 (SAP4). Par exemple winsock.dll est la bibliothèque dynamique qui permet d'utiliser TCP/IP sur une machine windows . En résumé un terminal offre trois points d'accès au réseau (Fig. 2) , un point (SAP2) au niveau de la ***carte*** réseau (ou du ***modem***), un point (SAP3) au niveau ***réseau*** (IP), un point (SAP4) au niveau ***applicatif*** (TCP). Si une application utilise un service au point SAP2 elle devra fournir une couche IP, ce qui veut dire en particulier posséder sa propre adresse IP. Une application qui s'interface au point SAP3 utilisera une adresse IP prédéfinie. Enfin une application indépendante des aspects réseaux (couche 3 et 4) utilisera le point SAP4.

Figure 2. Architecture de communication

L’architecture de communication d’une carte

Protocoles de transmission

Nous avons indiqué précédemment qu’une carte était reliée au monde extérieur grâce à une liaison série dont le débit est compris entre 9600 bauds et 105900 bauds. Le standard ISO 7816-3 [1] définit deux protocoles de transmission entre la carte et son lecteur.

- T=0 est un protocole orienté caractères, les octets sont transmis un par un, les erreurs sont détectées par un bit de parité et corrigées par retransmission.
- T=1 est un protocole orienté bloc, les octets sont transmis dans des trames, l’intégrité des données est vérifiée au moyen d’un CRC, la correction des erreurs est obtenue par retransmission de l’information.

Nous considérerons que les protocoles 7816-3 sont identiques à une couche OSI 2, dite couche de liaison de données, parce qu’ils sont responsables de la détection et de la correction des erreurs. En conséquence nous admettrons que le lien entre carte et lecteur est une liaison point a point, sans erreurs puisque ces dernières sont corrigées par les protocoles 7816.

Application Protocol data Unit (APDU)

Les couches OSI 5 (session) et 7 (application) ont été combinées en une seule entité définie par le standard ISO 7816-4 [1]. Une application logée dans une carte communique au moyen d’APDUs qui sont transportées par la ligne série. Une APDU contient un message de commande ou un message de réponse qui est envoyé ou reçu vers/depuis le lecteur. Le dialogue se déroule selon un paradigme du type question/réponse. Le terminal émet une question (.command) à laquelle la carte répond (.response). de fait la norme 7616-4 fixe les règles du dialogue entre la carte et le terminal de manière analogue à une couche session. Une commande APDU comprend au minimum 4 octets nommés CLA INS P1 P2. Les deux premiers octets (CLAss, INStruction) indiquent la nature de la commande (par exemple lecture, écriture). Les deux suivants (P1,P2) définissent des paramètres additifs de la commande (par exemple une adresse mémoire). La réponse comporte des octets de donnée optionnels et se termine par deux octets SW1 SW2 qui notifient le status de l’opération (succès ou échec). Ainsi le status 90 00 signifie qu’une opération s’est déroulée sans problèmes. Il existe quatre types d’APDUs selon que la commande ou la réponse soit associée à des données.

Echange de données asynchrone

La norme ETSI GSM 11.11 ([3] carte SIM) a notifié un mécanisme pour définir la longueur d’une réponse qui n’est pas connue à priori. La réponse à la commande est déterminée par l’interprétation des octets de status. SW1 sera égal à 9F et SW2 indiquera dans ce cas la taille des données attendues. Par exemple à la suite d’un ordre entrant la carte répond par deux octets de status 9F xx, ou xx est la longueur de la réponse. Le terminal émet alors une commande *GET RESPONSE* (A0 C0 00 00 xx) pour recueillir l’information. Un mécanisme similaire peut être utilisé pour échanger des données entre une carte et un terminal. Le terminal envoie des données vers la carte au moyen d’une commande d’écriture. La carte répond par un 9F xx, le terminal utilise alors une commande de lecture pour obtenir l’information disponible sur la carte.

Utilisation des APDUs

Côté carte une entité parfois appelée APDU Manager analyse le flux des APDUs entrantes et réalisent les opérations nécessaires. Par exemple dans une carte multi applications, une APDU *SELECT* est utilisée pour sélectionner une application particulière (un programme écrit en java, ou encore un cardlet). Cette opération étant réalisée, toutes les APDUs reçues sont routées vers l’application en cours. Dans un terminal un logiciel particulier gère le lecteur de cartes. En 1996 un ensemble d’industriels de la carte à puce et de systèmes informatiques ont adopté le standard PC/SC [5] qui permet l’intégration de lecteurs et de cartes dans les machines informatiques.

Une application émet/reçoit des APDUs vers/depuis un lecteur à l'aide d'APIs (Application Programmatic Interface) délivrées par le système d'exploitation. D'autres architectures sont également en cours de définition pour des plates-formes diverses (citons par exemple MUSCLE [7], OCF [6]...). Nous appellerons *AMUX* (APDU multiplexeur) la couche logicielle logée sur le terminal ou la carte qui réalise la commutation des APDUs vers les applications utilisatrices.

■ Architecture d'une carte à puce réseau

L'architecture que nous proposons est illustrée par la figure 2. Nous avons introduit une nouvelle couche (Smart Layer) qui utilise les services de l'entité AMUX. Il existe une instance de cette couche côté terminal (**H**ost **S**mart Layer B HSL) et une autre côté carte (**C**ard **S**mart **L**ayer - CSL). HSL accède aux ressources réseaux (SAPx) qui sont disponibles sur le terminal, et également aux APIs associées au lecteur de carte. Elle peut émettre ou recevoir des paquets vers / depuis le réseau. Elle établit un chemin logique entre des applications réseaux telles que navigateur web ou courrier électronique et une carte à puce insérée dans le lecteur. CSL voit le réseau grâce aux informations qui sont échangées avec HSL. Une couche Smart Layer comporte deux sous ensembles :

- Des Smart Agents.
- Une entité Smart Transfert Protocol (SmartTP), qui se comporte comme un commutateur de paquets.

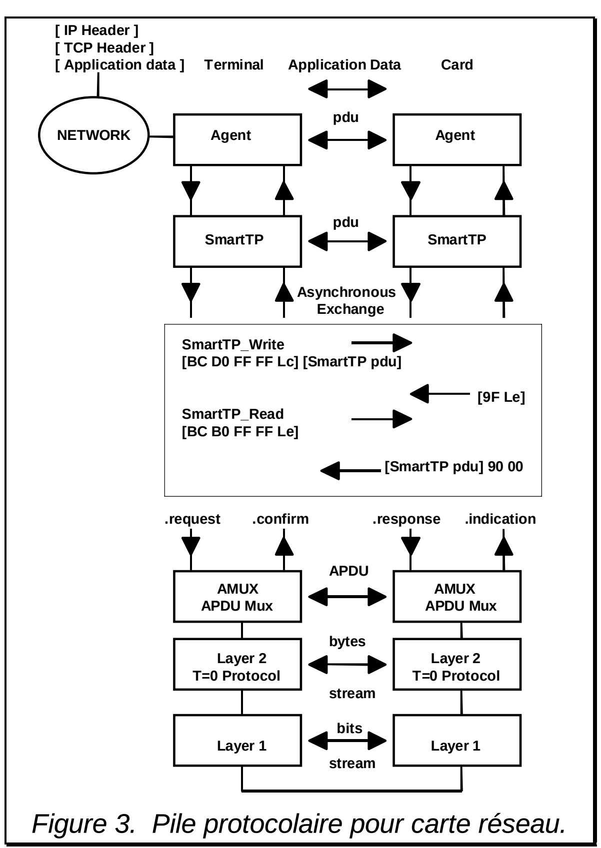


Figure 3. Pile protocolaire pour carte réseau.

Un Agent est une entité logicielle autonome (ELA), qui peut être réalisée sous diverses formes en fonction du système hôte, par exemple une DLL (Dynamic Link Library) dans un ordinateur personnel ou un Cardlet dans une carte à puce Java. Il est possible d'envisager le chargement dynamique d'agents en fonction des besoins du système. Un agent est identifié par une référence (un nombre de 16 bits) qui peut être fixe (*well known value*) ou éphémère. Côté terminal des agents réseaux sont capables d'utiliser les ressources réseaux du terminal et fournissent un accès internet aux agents cartes. On distingue trois types d'agents réseaux qui se différencient en fonction du SAP auquel ils ont accès :

- Les Agents réseaux ***applicatifs***, qui sont reliés à un SAP de niveau 4, et donc utilisent la totalité de la pile TCP/IP du terminal.
- Les agents réseaux ***raw***, dans ce cas une prise SAP3 est disponible et l'agent partage la couche IP du terminal.
- Les agents réseaux ***paquets***, qui ont accès à une prise SAP2, par exemple et sont capables d'émettre ou de recevoir des trames (par exemple des trames ethernet).

Les agents échangent des données au moyen de paquets que nous nommerons pdu (protocol data unit, SmartTP pdu en abrégé). L'entité SmartTP se comporte comme un commutateur qui gère le flux des *SmartTP pdu* échangé entre la couche AMUX et les agents.

Pile protocolaire de communication

La pile de communication utilisée par une carte réseau comporte cinq étages (Fig. 3).

- Une couche physique (OSI 1) qui assure l'échange de bits d'information (ISO 7816-2 & 7816-3).
- Une couche liaison de donnée (OSI 2) qui supporte les protocoles de transmission ISO 7816-3.
- Une entité AMUX qui route les APDUs depuis/vers une couche SmartTP.
- Une couche SmartTP qui assure l'interface avec les agents.
- Des agents qui traitent des données applicatives transportées par des SmartTP pdu.

La coopération de deux piles symétriques (une localisée dans une carte et l'autre dans le terminal associé) permet à un agent carte de traiter des protocoles (http 1.1 [14]) transportés par le réseau internet.

L'interface AMUX

Nous avons défini deux APDUs spécifiques (***SmartTP_Write*** et ***SmartTP_Read***) pour assurer le transport des *SmartTP pdu* au moyen du protocole de transmission T=0. Conformément aux principes de l'OSI l'interface logique avec l'entité SmartTP est un ensemble de quatre primitives (AMUX.request, AMUX.indication, AMUX.response, AMUX.confirm).

SmartTP pdu

Le protocole SmartTP est similaire à un protocole ***TCP light***. Le protocole TCP [9] définit la notion de numéro de port pour identifier les entités logicielles qui sont reliées par une connexion TCP. Une session TCP est gérée au moyen de six **flags** (URG, ACK, PSH, RST, SYN, FIN) qui sont utilisés pour l'ouverture et la fermeture d'une connexion et pour le contrôle de flux. Parce que SmartTP reçoit des données sans erreurs et dans l'ordre d'émission, nous avons importé de TCP seulement les notions de flag et de ports. Un *pdu SmartTP* est divisé en quatre sections :

- ***Source Reference*** est un champ de deux octets qui précise la référence de l'agent qui a émis le *pdu*.
- ***Destination Reference*** indique la référence de l'agent à qui est adressé le *pdu*.
- **Flag**, est un ensemble de 8 bits.
- **Data**, est un champ optionnel (de 0 à 256 octets) qui contient l'information échangée par les agents.

Flag est un ensemble de huit indicateurs (ou encore bits), dont trois ont une signification particulière.

- ***Open***, cet indicateur signifie que le *pdu* est associé à une ouverture de session entre l'agent source et destination.
- ***Close***, cet indicateur signifie que le pdu *associé* est le dernier d'une session.
- ***Block***, l'agent qui a émis le *pdu* suspend toute opération et restera dans un état d'*attente inactive* jusqu'à la réception d'un nouveau *pdu*.

D'autres indicateurs sont utiles pour de nombreux agents, par exemple cinq indicateurs (Open, Close, Block, Read, Write) sont suffisant pour reproduire le paradigme (OPEN-READ-WRITE-CLOSE) qui est utilisé dans les systèmes Unix pour la manipulation des fichiers. Un jeton (***token***) est un *SmartTP pdu* qui ne comporte pas de données (et dont la taille est donc égale à 5 octets). Au contraire un ***Data pdu*** est un *SmartTP pdu* associé à des données**.**

SmartTP

L'entité SmartTP a pour mission de commuter les pdu depuis/vers les agents. Par convention elle est associée à une référence nulle. Un jeton émis par SmartTP (***Null Token***) possède une référence source égale à zéro. L'aiguillage des pdu respecte quatre règles :

- SmartTP (côté carte) émet toujours un pdu après la réception d'un pdu. Cet élément est un ***Null Token*** ou bien est produit par un agent (*la carte répond toujours*).
- SmartTP (côté terminal) ignore les *Null Token*, il ne produit aucun pdu lors de la réception d'un tel élément.
- SmartTP génère un *Null Token* lors de la réception d'un pdu avec l'indicateur *Open* non positionné et dont la référence de destination est inconnue (*tolérance aux pdu perdus*).
- SmartTP produit un jeton avec un indicateur *Close* positionné lors de la réception d'un pdu avec l'indicateur *Open* positionné et dont la référence de destination est inconnue (*gestion des agents absents*).

Les Agents

Deux agents sont reliés au moyen d'un session**.** Un agent est identifié par un nombre de 16 bits (compris entre 0 et 65535). Le bit le plus significatif (b15) indique si l'agent est local (logé dans la même smart layer) ou dis-

tant (b15=0). Cette fonctionnalité sous entend que deux agents peuvent communiquer à l'*intérieur* d'un même système (carte ou terminal) ou entre un terminal et une carte. Six propriétés particulières caractérisent un agent :

- ***Terminal***, un agent logé dans le terminal.
- ***Carte***, un agent qui réside dans une carte.
- ***Local***, un agent qui n'accède pas à une ressource réseau.
- ***Réseau***, un agent qui accède à des ressources réseaux.
- ***Serveur***, un agent qui reçoit une demande d'ouverture de session (un SmartTP pdu avec l'indicateur *Open* positionné).
- ***Client***, un agent qui initialise l'ouverture d'une session (émission d'un SmartTP pdu avec l'indicateur *Open* positionné).

Côté terminal les agents réseaux sont capables d'accéder au web (utilisation de librairies réseau...). Ce sont les composants clés de notre architecture, d'un point de vue TCP ils se comportent comme des clients ou des serveurs, et offrent aux agents cartes auxquels ils sont connectés un point d'accès au réseau.

Les Sessions

Une session est un lien logique entre deux agents identifiés par une référence. Un agent client émet un SmartTP pdu dont l'indicateur *Open* est positionné. La référence d'un client est éphémère, elle est allouée par l'entité SmartTP au moyen d'une primitive GiveReference(), par exemple à l'aide d'un algorithme du type LRU (Last Recently Used). Un agent serveur reçoit un SmartTP pdu avec l'indicateur *Open* positionné. La référence d'un serveur est fixe, et connue par convention (*well known value*). Une session est identifiée par le couple des références client et serveur dont la valeur est pseudo unique (c'est-à-dire unique durant un intervalle de temps suffisant). Un SmartTP pdu dont l'indicateur *Close* est positionné est la marque d'une fin de session. Un client notifie à l'entité SmartTP la fin d'une session au moyen d'une primitive ReleaseReference() qui entraîne l'abandon de la référence éphémère.

Etats et règles associés à un Agent

Schématiquement un agent possède trois états:

- ***Déconnecté***, aucune session n'est en cours avec un autre agent.
- ***En Attente et Connecté***, une session est ouverte avec un autre agent.
- ***Bloqué***, l'agent est en attente d'un SmartTP pdu, et suspend toutes autres opérations (réseaux en particulier).

Un agent respecte ***quatre règles*** :

- Un agent ignore (il ne génère pas de pdu) tout SmartTP pdu dont l'indicateur *Open* n'est pas positionné, et qui n'est pas associé à une session active (*tolérance aux pdu perdus*).
- Un agent ne répond jamais (il ne produit pas de pdu) à un pdu dont l'indicateur *Close* est positionné. Si nécessaire un jeton est généré par l'entité SmartTP.
- Lorsqu'un agent reçoit un SmartTP pdu dont la référence de source est nulle et avec l'indicateur *Close* positionné, il referme toutes ses sessions actives.
- Un agent répond ***toujours*** à un SmartTP pdu dont l'indicateur *Block* est positionné, si nécessaire il produit un jeton.

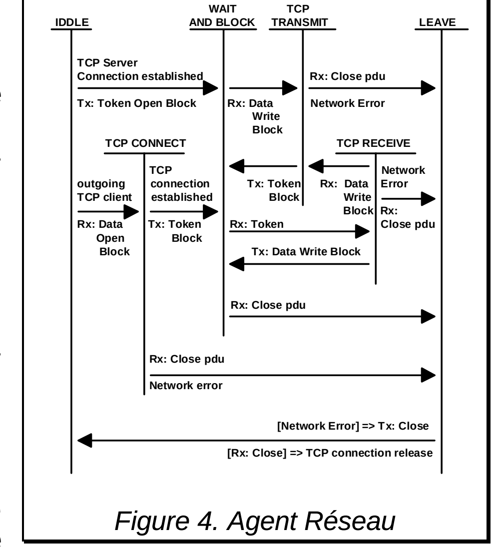


Figure 4. Agent Réseau

Les agents réseaux

Un agent réseau est la clé de voûte d'une carte à puce réseau. Il se comporte comme un proxy entre une connexion TCP/IP et un autre agent réseau. En raison des propriétés intrinsèques du protocole TCP il est nécessaire de définir deux sortes d'agents réseaux, les agents *TCP serveur* et les agents *TCP client*. Un agent TCP serveur transforme une connexion entrante TCP en un pdu d'ouverture de session (*Open*), un agent TCP client transforme un pdu d'ouverture de session (*Open*) en une connexion TCP vers un serveur (web) distant. Lorsqu'une connexion TCP a été établie et qu'une session inter agent lui a été associée, un tunnel est ouvert entre un agent carte et le web (par le biais des services d'un agent réseau). Nous allons à présent décrire l'agent réseau que nous avons réalisé. Ce dernier comporte six états, et est organisé autour d'un pivot central (WAIT_AND_BLOCK) pour lequel aucune opération n'est menée. L'agent est en attente d'un SmartTP pdu qui déterminera son prochain état (lecture ou écriture réseau).

- ***IDDLE***, dans cet état un serveur TCP est en attente d'une connexion, ou bien un agent client attend un SmartTP pdu *Open*.
- ***TCP_CONNECT***, cet état n'a de sens que pour un client TCP. Un SmartTP pdu dont l'indicateur Open est positionné a été reçu. Les données associées spécifient le port et l'adresse d'un serveur web distant. Une connexion TCP est en cours d'ouverture.
- ***WAIT_AND_BLOCK***, l'agent attend la réception d'un SmartTP pdu, toutes les opérations réseaux sont gelées.
- ***TCP_TRANSMIT***, des données sont en cours d'émission sur le réseau internet.
- ***TCP_RECEIVE***, l'agent est attente de réception d'information depuis le réseau internet.
- ***LEAVE***, si une erreur réseau s'est produite un *Token Close* est émis, dans le cas contraire l'agent ferme la connexion TCP.

■ La carte à puce serveur web

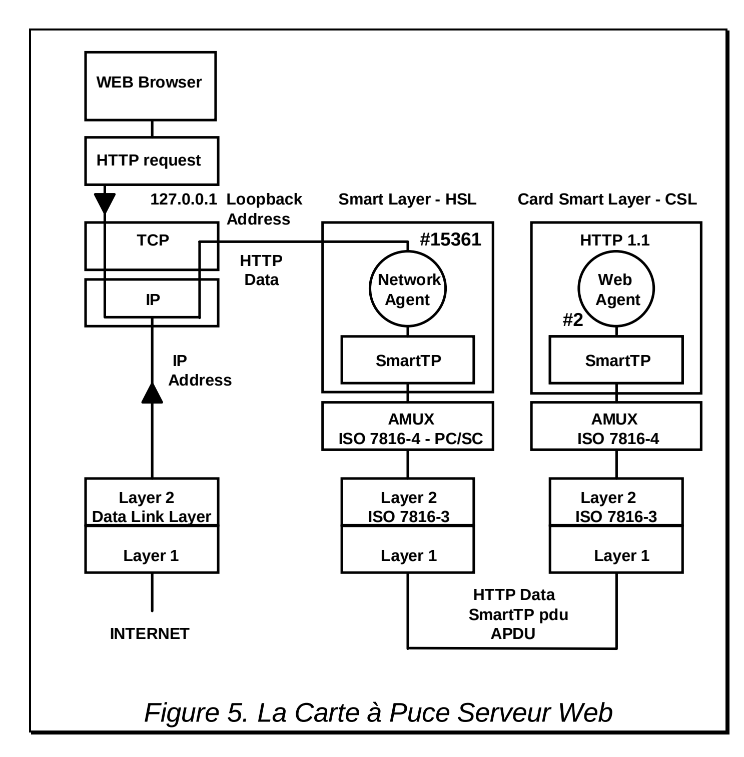


Figure 5. La Carte à Puce Serveur Web

Un serveur web est un protocole définit par un standard RFC (Hyper Text Transfert Protocol RFC 2068 HTTP 1.1). Son implémentation dans une carte à puce se traduit par le fait que les données http, échangées à travers le web par des paquets TCP/IP, sont transportées entre la carte et le terminal par des SmartTP pdu. D'un point de vue logique une session est ouverte entre un client (le navigateur) et un serveur web logé dans la carte.

L'ensemble internet-terminal-carte constitue un média qui véhicule les données applicatives (http). Une carte à puce traditionnelle est ***APDU centric,*** une ressource de la carte est accessible grâce à un jeu d'APDUs. Une carte réseau est ***URL centric***, c'est-à-dire que les ressources offertes sont mises en œuvre à partir d'URLs. Considérons à présent un terminal qui comporte une pile TCP/IP et un navigateur web. L'Url http://127.0.0.1:8080 dans laquelle 127.0.0.1 est l'adresse de boucle du terminal est 8080 le port TCP associé à un agent réseau (lui même connecté à un agent carte -serveur web), permet d'accéder à un index de la carte (une page html nommée par exemple index.html). Cette dernière comporte des hyperliens qui désignent des ressources externes ou internes. Nous appelons ***Terminal Virtuel*** le fait qu'à travers son serveur web la carte présente au terminal une représentation virtuelle de ses ressources. Une ressource carte peut être une entité cryptographique (algorithme divers...), un objet multimédia (page html, son , image) ou un logiciel (applet java...). Une carte internet est un dispositif ***ouvert*** qui peut être utilisé sans connaissance préalable (***zero knownledge device***). Un terminal TCP/IP possède deux adresses IP remarquables, une adresse dite de boucle (127.0.0.1) et une adresse internet véritable, en conséquence une carte internet peut être accédée de manière locale à partir de son terminal ou à distance depuis un point quelconque du web. De manière pratique, un navigateur ouvre simultanément plusieurs sessions TCP avec un serveur web (4 est une valeur courante), un agent réseau devra donc sérialiser ces différentes sessions, par exemple au moyen d'un sémaphore.

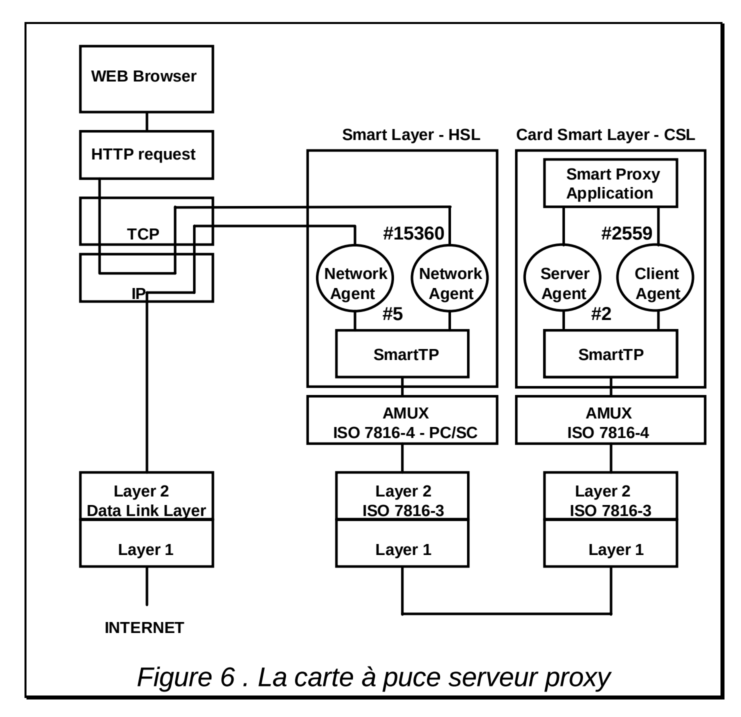


Figure 6 . La carte à puce serveur proxy

Une session http

Une session http (Fig. 5) se déroule en deux étapes, dans un premier temps une requête http est émise par le client et dans un deuxième temps le serveur répond par un entête qui précise la nature du fichier, suivi par le contenu du fichier proprement dit. Une particularité d'une carte internet est qu'elle doit être en mesure de fournir entête pour tous les fichiers qu'elle stocke. Nous allons à présent donner plus de précisions sur le déroulement d'une session http.

- Le navigateur ouvre une connexion TCP avec un agent réseau, une session est ouverte avec un agent carte, **T**[s=15361,d=2,open+block], **C**[s=2,d=15361].
- La requête http est transmise depuis l'agent réseau vers l'agent serveur web, **T**[s=15361,d=2,write+block,data], **C**[s=2,d=15361].
- La dernière partie de la requête est transmise**, T**[s=15361,d=2,write+block,data].
- Le serveur carte transmet le fichier et son entête**, C**[s=2,d=15361,write+block,data], **T**[s=15361,d=2,block], **C**[s=2,d=15361,write+block,data], **T**[s=15361,d=2,block].
- Le serveur carte met fin à la session**, C**[s=2,d=15361,write+close,data].

■ Un serveur proxy dans une carte à puce

Dans la technologie TCP/IP on nomme proxy une entité logicielle qui réalise d'une part un serveur TCP/IP et d'autre part un client TCP, créé de manière dynamique et qui réalise une connexion avec un autre serveur TCP distant. Le client se connecte avec un serveur distant prédéterminé ou déduit de manière dynamique en fonction des informations échangées sur la connexion établie avec le serveur. Un proxy réalise usuellement des fonctions de filtre (translation de protocole) et/ou de sécurité. Par exemple un proxy http assure généralement la connexion d'un navigateur à un serveur web dans une entreprise (pare-feu firewall), ou réalise des opérations de sécurité nécessaires (authentification B confidentialité B intégrité) à l'établissement d'un tunnel sécurisé à travers internet (Secure Socket Layer SSL) est bon exemple d'une tel filtre. Un proxy logé dans une carte à puce (Smart Proxy) associe à une url donnée une procédure particulière qui implique la connexion à un autre serveur externe. La procédure de connexion est réalisée par la carte et peut comporter des mécanismes d'authentification. Par exemple l'Url http://127.0.0.1:8080/eMail désigne un fichier eMail localisé dans une carte à puce internet. Le fichier eMail est en fait associé à une procédure de connexion sur un serveur distant dont la carte connaît le login et le mot de passe. Ce mécanisme permet de se connecter à de nombreux services gratuits disponibles sur internet. Nous appelons ***Fichier Virtuel*** un objet identifié par une url carte mais logé sur un serveur web externe. De manière plus générale un proxy carte à puce est un élément clé pour une nouvelle génération d'application qui utiliseront ce dispositif comme une passerelle de sécurité (*trusted proxy*). La carte agit comme un filtre actif entre l'utilisateur et le réseau, elle traite ou mémorise selon des règles prédéterminées les données entrantes ou sortantes. Une application *proxy carte* met en œuvre deux sessions simultanées et traitent les données reçues par deux agents réseaux (un serveur TCP et un client TCP).

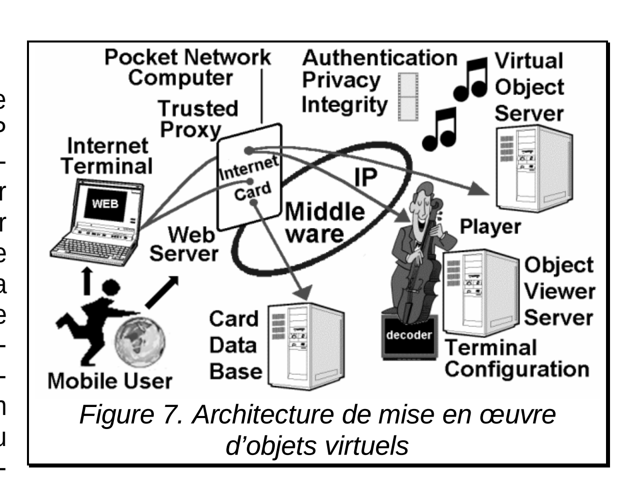


Figure 7. Architecture de mise en œuvre d'objets virtuels

■ Conclusion

Nous avons implémenté le concept de la carte à puce internet sur un carte Java Odyssey 1.2 dont la taille mémoire utilisateur est de 7 ko. Le code Java qui réalise un serveur web, un smart proxy, et un système de gestion de fichier est d'environ 3.5 ko. La mémoire allouée aux fichiers est donc de 3.5 ko. Des fichiers classiques (pages html, images, applets) ont été stockés dans la carte ainsi que des fichiers virtuels. Il est possible de protéger certain fichiers par un verrou, chaque fois que le navigateur essaye d'accéder à un fichier verrouillé le serveur de la carte répond par un formulaire qui demande à l'utilisateur de fournir un mot de passe. Un smart proxy associe à un *fichier virtuel* nommé eMail une procédure de connexion à un serveur de courrier gratuit auquel la carte fournit un login et un mot de passe encapsulés dans un formulaire http (POST). Le résultat est que le porteur de la carte peut lire son courrier électronique sans connaître aucune information sur le serveur qui l'abrite. La couche HSL a été écrite pour un ordinateur personnel équipé d'un système d'exploitation win32. La taille du logiciel est modeste de l'ordre de 100 ko. Les performances en termes de débits sont comprises entre 80 et 200 octets/seconde en fonction du type de lecteur de carte utilisé (soit une fourchette moitié, de 40 à 100 octets/s pour les applications smart proxy). Nous avons donc démontré qu'il est possible de réaliser des cartes internet à partir de cartes Java existantes, capables de s'adapter à des services déjà présent sur le web. Nous pensons que cette nouvelle technologie est une première étape vers la définition d'applications réseaux qui mettrons en œuvre des *objets virtuels*. Les puces disponibles au début du prochain millénaire seront à base de processeur RISC 32 bits et auront la puissance nécessaire pour offrir la sécurité (authentification, chiffrement, intégrité) et assurer l'usage de tels objets. En conclusion nous présentons (Fig. 7) l'ébauche d'une architecture de mise en œuvre d'objets virtuels, elle comporte schématiquement cinq éléments.

- Un *utilisateur mobile*, propriétaire d'objets virtuels, qui dispose d'une pluralité de terminaux internet.
- Une *carte à puce internet*, qui se présente de manière logique comme un serveur web et comporte un ou plusieurs smart proxy de confiance (*trusted proxy*), accède à diverses applications grâce à un bus de communication internet (Middleware IP), qui utilise différentes méthodes de communication.
- La carte est associée à une *base de donnée* qui abrite une partie des informations détenues par son propriétaires. Les accès à ce serveur sont sécurisés par la carte internet.
- A partir des données fournies par sa base de donnée l'utilisateur réalise l'image d'un *objet virtuel* (stocké sur un serveur) sur le terminal internet. Cet objet peut être associé à divers aspects sécuritaires (authentification, chiffrement, intégrité...).
- Il peut être nécessaire de configurer le terminal avec des logiciels spécifiques (*Object Viewer*) associé à l'usage de certain objets. Dans ce cas la carte fait office de *licence mobile* d'un tel logiciel.

■ Références

1. International Organization for Standardization, "Identification Cards - Integrated Circuit(s) Cards with Contacts", ISO 7816.
2. International Organization for Standardization, "Contactless integrated circuit(s) cards - Proximity Cards", ISO 14443.
3. European Telecommunications Standards Institute, "Digital cellular telecommunications system (Phase 2+) Specification of the Subscriber Identity Module - Mobile Equipment (SIM - ME) interface" ETSI GSM 11.11
4. European Telecommunications Standards Institute, "Digital cellular telecommunications system (Phase 2+) Specification of the SIM Application Toolkit for the Subscriber Identity Module - Mobile Equipment (SIM -ME) interface", ETSI GSM 11.14.
5. "Interoperability Specification for ICCs and Personal Computer Systems", PC/SC, © 1996 CP8 Transac, HP, Microsoft, Schlumberger, Siemens Nixdorf.
6. OpenCard Framework - OCF - http://www.opencard.org.
7. Movement for the Use of Smart Cards in a Linux Environment.- MUSCLE - http://www.linuxnet.com/smartcard/index.html.
8. International Organization for Standardization, "Information Processing Systems - Open Systems Interconnection - Basic Reference Model ", ISO 7498.
9. J.Postel, "Transmission Control Protocol", Request For Comment RFC 793, September 1981.
10. "Le musée de la carte" - http://www.cardshow.com/museum/exposition.html
11. F.Guez C.Robbert A.Lauret - "Les cartes à microcircuit" - MASSON 1988
12. Louis Claude Guillou, Michel Ugon, and Jean-Jacques Quisquater, "The Smart Card: A Standardized Security Device Dedicated to Public Cryptology,", Contemporary Cryptology. The Science of Information Integrity, ed. Gustavus J.Simmons, IEEE Press 1992, pp. 561-613
13. R.Merckling, A.Anderson, "Smart Card Introduction", Request For Comment RFC 57, March 1994
14. T.Berners-Lee & All, "Hypertext Transfer Protocol - HTTP/1.1", Request For Comment, RFC 2068, January 1997.
15. Robert W.Baldwin, C.Victor Chang, "Locking the e-safe", IEEE Spectrum, February 1997, pp 40-46.
16. Carol Hovenga Fancher, "In Your Pocket: Smartcards", IEEE Spectrum, February 1997, pp. 47 - 53.
17. MichaelC.McChesney, "Banking in cyberspace: an investment in itself. ", IEEE Spectrum, February 1997, pp. 54 - 59.
18. Group D, I.S.206, SIMS - "Future Organisation of the Computer and Communications Industries" - http://www.sims.berkeley.edu/courses/is206/f97/GroupD/dfutre.html - 1997

19. Constantinos Markantonakis, "The Case for a Secure Multi-Application Smart Card Operating System", Information Security Workshop 97 (ISW'97), September 1997 (Ishikawa in Japan), Springer-Verlag (LNCS 1396), Berlin (1997), pp.188-197.
20. Roger Dettmer, "Getting Smarter", IEEE Review, pp. 123 - 126, May 1998.
21. Ton Verschuren, "Smart access: strong authentication on the web", Computer Networks and ISDN Systems, 30, 1998, pp 1511-1519
22. ISI - "IBM Smart Card Identification Protocol" - http://www.iscit.surfet.nl
23. Jim Walejko "SmartCard Architecture" - NTU CA714-CA Graduate Computer Architecture - http://www.cs.berkely.edu/~neefe/ntu.fa98/jim.project.html - 1998
24. Naomaru Itoi, Peter Honeyman, "Smartcard Integration with Kerberos V5", Proceedings. of USENIX Workshop on Smartcard Technology, Chicago, May 1999, pp 51-62
25. Naomaru Itoi, Peter Honeyman, Jim Rees, "SCFS: A UNIX Filesystem for Smartcards", Proceedings of USENIX Workshop on Smart Card Technology, Chicago May 1999, pp 107-118.
26. Bruce Schneier, Adam Shostack, "Breaking Up Is Hard to Do: Modeling Security Threats for Smart Cards", Proceedings of USENIX Workshop on Smart Card Technology, Chicago May 1999, pp 175-185.
27. ACTIV CARD - "ActivCard Synchronous Authentification - A white paper" - http://www.activecard.com/pressrom/whitepapers/des.html
28. Microsoft - "SmartCard White paper" - http://www.microsoft.com/windowsce/smartcard/ 1999.